Glossary

Account Group: *A template for establishing a runtime environment context for individual operators*. Account groups are typically used to do a high-level segregation of operators into system administrators, security administrators, database administrators, or mission-specific operators.

Advanced Interoperability: *A level of interoperability characterized by shared data between applications, including shared data displays, and information exchange through a common data model*. This level provides for sharing of information in a distributed, but localized environment, and for sharing of applications.

Affected Account Group(s): *The account group(s) to which a segment applies*. Functionality provided by the installed segment will normally appear to the operator as new menu items or icons in the affected account group(s).

Aggregate Segment: *A collection of segments grouped together, installed, deleted, and managed as a single unit*. Aggregates are a convenient way for grouping segments that need to be developed and managed separately, but which must be presented to an operator as a single collection of functions.

Application Programmer Interface (API): *A programmer's guide that describes the data structures, function calls, and services provided by the COE, and how to write software modules that interface with and use COE services*. The definition of an API is programming-language neutral and includes traditional function (C/C++) or procedure (Ada) interfaces as well as command-line interfaces. Data structures are included in the definition as are the definition of objects for services that are written in an object-oriented model.

Approved Software: *Software that has been tested as compatible with the COE*. An approved products list might contain Oracle, Sybase, WordPerfect, Kermit, etc. In this context, approved software implies only that the software has been tested and confirmed to work within the DII COE. It does *not* imply that the software has been approved or authorized by any government agency for any specific system.

Architecture: *The structure of components, their relationships, and the principles and guidelines governing their design and evolution over time* (IEEE STD 610.12). Three types of architectures are defined: operational, technical, and systems.

Assigned Directory: *The unique directory assigned to a segment at segment registration time that is to be the segment's home directory*. A segment is normally completely contained within its assigned directory and may not modify any files outside its assigned directory without going through facilities provided by the COE.

Basic Interoperability: *A primitive level of interoperability characterized by peer-to-peer connected systems that allow basic exchange of homogenous data (e.g., email, formatted messages), and allows for basic collaboration*. This level of interoperability is

achievable by relatively simple interfacing techniques, and by use of standard office automation products that provide data import/export functions for handling data from another product.

Bootstrap COE: *The subset of the kernel COE that is loaded in order to have enough of an operational environment that segments can be loaded*. The bootstrap COE is typically loaded along with the operating system through vendor-supplied instructions or low-level Unix commands such as `tar` and `cpio`. The specific contents of the bootstrap COE will usually differ from platform to platform because it depends upon the techniques provided by the vendor for loading initial software and data.

Broker: *An intermediary that coordinates and manages the requests between clients and servers*. Brokers are generally discussed in the context of CORBA (Common Object Request Broker Architecture) or other object-oriented approaches, but the usage need not be so restrictive. An example of a broker operation is to connect a client requesting some service to an available server located at some arbitrary location on the LAN.

Client: *A computer program, such as a mission application, that requires a service*. Clients are consumers of data while servers are producers of data.

Client/Server: *A particular kind of computing architectural model in which consumers (clients) and producers (servers) cooperate to create an application*. Clients request services from servers, while servers may service one or more clients simultaneously. A typical example of a server is a database server. An example of a client is a software component that allows an operator to prepare a query, pass the query to the database, and then display the results to the operator.

COE-Component Segment: *A segment that is contained within the COE*. All software in COE-based systems is packaged as a segment, including those within the COE itself. Strictly speaking, "COE component" is a segment attribute rather than a separate segment type. Segments are specifically identified as COE components because specialized processing is performed on them during software installation, and they are handled more rigorously in the development cycle.

Commercial Off-The-Shelf Software (COTS): *Software that is available commercially*. Examples include versions of Unix, X Windows, or Motif, as well as standard products such as Oracle, Sybase, and Informix.

Common Operating Environment (COE): *The collection of standards, specifications, and guidelines, architecture definition, software infrastructure, reusable components, APIs, methodology, runtime environment definition, reference implementation, and methodology that establishes an environment on which a system can be built*. The COE allows segments created by separate developers to function together as an integrated system. The COE is the vehicle that assures interoperability through a reference implementation that provides identical implementation of common functions. It is important to realize that the COE is both a standard and an actual product (e.g., reference

implementation) composed of reusable software components built according to a set of open standards and specifications.

Community Files: *Files that reside outside a segment's assigned directory*. To prevent conflict among segments, community files may be modified only through the COE installation tools. Examples of community files include `/etc/passwd`, `/etc/hosts`, and `/etc/services`.

Compliance: *An integer value, called the compliance level, which measures (a) the degree to which a segment or system achieves conformance with the rules, standards, and specifications described by the COE, (b) the degree to which the segment or system is suitable for integration with the DII COE reference implementation, and (c) the degree to which the segment or system makes use of COE services*. Compliance is measured for segments and COE-based systems. It is important to note that compliance is a matter of degree. It is usually not an "all or nothing" proposition, but it is possible to fail to meet any COE objectives and hence be declared totally non-compliant. Compliance is measured in four areas, called compliance categories. The four categories are Runtime Environment, Architectural Compatibility, Style Guide, and Software Quality. The higher the level of compliance, the higher the level of interoperability with other COE-based systems.

Compliance Category: *One of the four areas (Runtime Environment, Style Guide, Architectural Compatibility, Software Quality) in which DII compliance is measured*. These four categories form a spectrum for measuring compliance with regard to COE compatibility and degree of interoperability (Runtime Environment), user friendliness (Style Guide), product longevity (Architectural Compatibility), and program risk (Software Quality).

Compliance Level: *The degree to which a segment is DII-compliant within a specific compliance category*. Compliance levels are integer values only.

Component Database: *Individual database within a multi-database design*.

Composite Compliance: *The DII compliance value assigned to a collection of segments*. The purpose of a composite compliance value is to describe the degree of compliance a system achieves when it may contain COE-component segments that themselves are not Level 8 compliant. The composite compliance level for an arbitrary collection of segments is the compliance level of the *least* compliant segment. Chapter 2 provides formulas for computing the composite compliance level for a COE-based system, and for systems which contain both COE and non-COE based platforms.

Configuration Control Board (CCB): *The organization responsible for authorizing enhancements, corrections, and revisions to the COE, or to a COE-based system*. CCBs exist for the purpose of controlling changes made to a system. DISA chairs the CCB for the DII COE and for its own systems (e.g., GCCS, GCSS, ECPN). Other services and agencies may use the same approach for controlling changes to their COE-derived systems.

Configuration Definition: *A hierarchical representation of a collection of segments that are to be installed*. Configuration definitions are organized into distributions, folders, configurations, bundles, and segments. The purpose of configuration definitions is to allow pre-definition of the segments that are to be installed at a site broken down into whatever groups are meaningful to the site (e.g., workspace, workstation usage).

Data Store: *A functional grouping of data storage based on the type of information, or use of the storage*. A data store in Sybase is a database partition, while in Oracle it is a tablespace.

Database: *A structured set of data, managed by a DBMS, together with the rules for accessing the data*. A database is more than just a collection of data files.

Database Administrator: *The DBMS user account for DBMS administration functions*. Sybase uses "`sa`" for the DBA account while Oracle uses "`system`."

Database Management System (DBMS): *Software to manage concurrent access to shared databases*. There are several techniques for organizing and managing databases. Most commercial products are relational database management systems but new technology advances are making object-oriented database management systems a reality.

Database Owner: *The DBMS user account that is the creator or owner of the data objects that are part of the data store segment*.

Database Schema: *The specific data view or design of a particular database*. One of the powerful features of a database system is that different applications can have their own view of the database by defining different schema. The database management software handles the details of mapping the actual data representation on disk into the view the application requires.

Database Segment: *A standard method for packaging a physical database for incorporation into the COE/SHADE*. Database segments are packaged like any other COE segment.

Database Services User: *A special DBMS user account accessed only by an autonomous application such as a message processor that provides services to other users via the DBMS*.

Database Session: *An individual connection between an application program and a database management system*. Sessions are defined as a security measure for database accesses.

Descriptor Directory: *The subdirectory* `SegDescrip` *associated with each segment*. This subdirectory contains descriptors that provide information required to install the segment.

Descriptors: *Data files (contained in the segment's descriptor directory) that are used to describe a segment to the COE*. The software installation and integration process uses descriptor directories and their descriptor files to ensure DII compliance. Descriptor files permit automated integration and installation and are the technique for segments to "self-describe" themselves to the COE.

Development Environment: *The software environment required to create, compile, and test software*. This includes compilers, editors, linkers, debug software, and developer configuration preferences such as command aliases. The development environment is distinct from the runtime environment, and **must** be separated from the runtime environment, but it is usually an extension of the runtime environment.

Distributed Database: *A database whose data objects exist across, or are fragmented across, multiple computer systems or sites*. The ability to distribute a database across multiple sites is key to collaborative planning in a number of problem domains including GCCS, GCSS, and ECPN.

Distributed Processing: *The ability to perform collaborative processing across multiple computers*. Among its many benefits, this capability allows processing load to be distributed across computers within the network. It also allows applications running on one platform to access capabilities that it may not have the hardware resources to host itself. DCE, CORBA, and DCOM are all techniques for performing distributed processing.

Environment: *In the context of the COE, all software that is running from the time the computer is rebooted to the time the system is ready to respond to operator queries after operator login*. This software includes the operating system, security software, installation software, windowing environment, COE services, etc. The environment is subdivided into a runtime environment and a software development environment.

Environment Extension: *The process of a segment issuing requests to the COE tools to expand the environment*. The principle of environment extension is important because it allows for segments to be easily added or removed, and it eliminates unarbitrated environmental conflicts between segments.

Environment Extension File: *A file that contains runtime environmental extensions for the COE*. Segments use extension files to add their own environment variables to the runtime environment, additions to the X Windows environment, etc. Environment extension files are the technique for encapsulating a segment's contribution to the runtime environment so that its environment effects can be easily added or removed.

Fragmentation Schema: *The distribution design for a distributed database*. The idea is that a schema (e.g., an application's view of the database) may actually require access to data that is distributed across computers or networks.

Government Off-The-Shelf (GOTS) Software: *Software developed through funding by the US Government*. The DII COE contains both COTS and GOTS segments.

Integration: *The process of combining components, usually hardware and software, into a new, larger component to achieve some architectural requirement*. Integration requires resolution of compatibility issues between components that are to be interconnected. Integration attempts to allow sharing of a common resource (such as data) without the need for intermediate translations from one format to another. Note that the COE is a technique for achieving integration that ensures interoperability.

Interfacing: *The process of two components or systems exchanging information by first translating the information into an intermediate, agreed-upon format*. For example, track information is transmitted between FOTC participants by generation of an OTH-GOLD message on the sending system, and parsing the message on the receiving system. If the two systems were truly integrated, the translation to/from OTH-GOLD would not be required. Note that standards are the technique for specifying how interfacing can be accomplished.

Intermediate Interoperability: *A level of interoperability characterized by a client/server environment with standardized interfaces and distributed computing services that allow for exchange of heterogeneous data (e.g., maps with overlays, annotated images), and advanced collaboration*. This level of interoperability is achievable with implementation of "cut and paste" between applications, through World-Wide-Web technology, and through basic use of DII COE features.

Interoperability: *The ability of two or more systems or components to exchange and use information* (IEEE STD 610.2). This definition is extended in the context of a COE to include levels of interoperability, and relate interoperability to interfacing (lowest, least desirable level) versus true integration (highest, most desirable level).

Kernel COE: *That subset of the COE-component segments that is required on all workstations*. As a minimum, this consists of the operating system, windowing software, security, segment installation software, and an Executive Manager. The DII COE is designed to minimize the size of the kernel so that minimal resources are required at each workstation. Definition of the kernel is independent of whether the workstation will be used as a database server, an applications server, or a client workstation.

Mission-Application Segment: *A segment that uses the services of the COE to provide functionality that is specific to a mission domain*. Mission-application segments are external to the COE and are built on top of the COE to create a new capability.

Multi-Database: *A collection of autonomous databases*. The databases may exist on a single platform, or be distributed across multiple platforms.

Open System: *A system that implements sufficient open specifications for interfaces, services, and supporting formats to enable properly engineered applications software: (a) to be ported with minimal changes across a wide range of systems, (b) to interoperate with other applications on local and remote systems, (c) to interact with users in a style that facilitates user portability, and (d) to enable users to increase processing power as their functional needs grow, without the need to re-write applications (i.e., scalability)*

(POSIX ISO 9945-1). The "openness" of a system is a multi-dimensional measurement that in essence determines how easy it is to change the system from one intended purpose to another.

Operational Architecture: *Descriptions of the tasks, operational elements, and information flows required to accomplish or support a warfighting function. The operational architecture is best thought of as that which identifies the warrior's need* (*JTA*). The operational architecture is closely tied to CONOPS (Concept of Operations) and is often dictated by policy rather than technology. It represents the desired system process and is more of the "what" rather than the "how." An example of an operational architecture is the reporting chain of command from a soldier in the field to the CJTF.

Plug and Play: *The ability for a COE-derived system to be scaled by the addition/removal of hardware or software components*. To have the property of "plug and play," the system must be able to reconfigure itself automatically after component addition/removal with required human intervention limited to no more than a system power down or reboot.

Portability: *The degree to which system components may be transferred from one hardware or software environment to another* (IEEE STD 610.12). High portability means that the transfer occurs with minimal or no modifications. For example, a COE segment which was originally developed on a Solaris platform that can be executed correctly on a Hewlett Packard or Silicon Graphics platform by rebuilding the segment without modifying the source code is highly portable.

Profile: *The subset of the total COE-based functionality that is to be made available to a group of individual operators*. Profiles are defined by the security or system administrator to limit what an operator can access both on a "need to know" basis and as a way to avoid overwhelming the operator with functions that are not useful for his mission area.

Reference Implementation: *The set of software and hardware components that implement a COE*. The reference implementation of a COE may change over time to take advantage of new technologies or to fix problem reports, but the principles governing the COE remain unchanged. A COE is incomplete without a reference implementation because it is the reference implementation that is most responsible for ensuring interoperability. The DII COE is an example of a reference implementation.

Remote Install: *The ability to electronically install segments from a local site (such as the DISA Operational Support Facility) to a remote site (such as USACOM)*. In a "push" mode, the local site initiates and controls the segment installation. In a "pull" mode, the remote site initiates and controls the segment installation.

Reusability: *The ability of a software or hardware component, built to fulfill a requirement for a system built to achieve a particular mission, to be used to meet a similar requirement for another system that is being built to address a different mission need*. Reusability allows system development and maintenance costs to be reduced

because they can be shared across multiple programs. The DII COE contains a reuse strategy.

Runtime Environment: *The runtime context determined by the applicable account group, the COE, and the executing segments*. The runtime environment definition is limited to just the context (environment variable settings, X configuration, background processes, task priorities, etc.) required for an application to execute. Environment settings for software development purposes are maintained separately from the runtime environment to avoid runtime conflicts between developer preference settings.

Scalability: *The ability of a system to increase (decrease) functionality , the amount of data which can be processed or stored, and responsiveness without the need to re-write applications through the addition (removal) of hardware and software components*. Scalability is critical in achieving "NCA to foxhole" at reasonable cost.

Segment: *A collection of one or more CSCIs most conveniently managed as a unit of functionality*. Segments are defined from the perspective of an operator, not a developer, and are generally defined to keep related CSCIs together so that functionality may be easily included or excluded. They are usually defined as functional pieces (e.g., a word processor) that make sense from a system administrator perspective because segments are the lowest level components that can be installed on, or removed from, a platform

Segmentation: *The engineering process of decomposing system components into segments and creating the appropriate segment descriptor files*. Proper segmentation is vital to a good system design and affects how well the component will operate in the resulting system.

Segment Compliance: *The degree to which a segment, whether it is part of the COE or a mission application, is compliant*. Compliance of individual segments is computed in order to calculate overall system compliance.

Segment Descriptor: *An ASCII file which contains information that describes attributes about the segment*. The format of these files is described in the *I&RTS*. Segment descriptor files are processed by various tools in the COE to validate compliance and to perform segment installation.

Segment Descriptor Directory: *The directory,* `SegDescrip`, *which contains the segment descriptor files used to describe a segment to the COE*. Each segment is required to contain a segment descriptor directory in order to be DII-compliant.

Segment Prefix: *A 1-6 alphanumeric character string assigned to each segment for use in naming public symbols*. Segment prefixes are used in any situation where it is important to make sure that there will not be a naming conflict between developers (environment variable names, executable file names, shared library names, etc.). Segment prefixes are not necessarily unique among all segments, but the combination of a segment prefix and segment name is *always* unique among all segments.

Segment Servers: *One or more designated workstations on a LAN that have segments stored on them in a format that can be used for installation on other workstations*. Designation of segment servers greatly simplifies distribution and installation of a system. It also speeds up installation because it can be done from disk across the LAN rather than from slower tape magnetic media.

Server: *A computer program that provides some service*. Servers are producers of data while clients are consumers of data.

Service: *A function that is common to a number of programs, such as performing some extensive calculation or retrieving a category of data*. An example of a service is a function that accepts a request to transform a point from one coordinate system into another.

Session: (See Database Session)

Shared Data Server: *A DII-compliant platform that provides data server functions for multiple mission applications.*

Shared Database Segment: *A database segment that supports the information requirements of multiple applications or across multiple database segments*. Shared database segments are typically mission-or-functionally-oriented, and are generally specific to a limited number of mission domains. Shared database segments are under joint configuration control. An example of such a database segment is a database of logistics drawings for military hardware.

Shared Data Environment (SHADE): *The DII COE strategy and mechanisms for data sharing*. SHADE includes the required data-access architectures, data sharing methodology, reusable software and data components, and guidelines, standards, and specifications for the development and migration of systems that meet the user's requirements for timely, accurate, and reliable data.

Superset: *The total collection of all COE-based segments available to the development community*. The superset includes the COE as well as mission-application segments.

System: *A set of different elements so connected or related as to perform a unique function not performable by the elements alone. The most important and distinguishing characteristic of a system, therefore, is the relationships among the elements*. An example is an automobile. All the individual elements may be in working order but do not individually provide transportation. Transportation only exists when all the elements are connected and function as a whole.

Systems Architecture: *Descriptions, including graphics, of systems and interconnections providing for or supporting warfighting functions* (*JTA*). The system architecture typically relates capabilities to requirements and requirements to standards.

System Compliance: *The degree to which a COE-derived system in total meets COE standards, specifications, and rules*. System compliance is a composite measure of all segments in the system whether they are part of the COE, or are application segments. The measure also includes provisions for components that are not COE-derived (e.g., mainframe components).

Technical Architecture: *A minimal set of rules that govern arrangement, interaction, and interdependence of the system components* (*JTA*). The technical architecture identifies standards and conventions to be used.

Unique Database Segment: *A database segment that is limited in scope and typically used by only one application*. Unique database segments may be shared between applications, but the usage is restricted to a single mission domain. An example of a Unique database segment is a configuration table that an application reads at initialization time. Unique database segments are under the configuration control of the segment sponsor.

Universal Database Segment: *A database segment that represents widespread data requirements and is used by many applications, many database segments, and spans multiple mission domains.* They are typically reference or lookup tables. An example is a database of country-code abbreviations. Universal database segments are under stricter configuration control with DISA and DOD Data Administration coordination.

Universal Interoperability: *An interoperability level characterized by the ability to globally share integrated information in a distributed information space*. Universal interoperability represents the ultimate interoperability goal.